下载APP

05 | 网盘系统设计：万亿 GB 网盘如何实现秒传与限速？

2022-02-25 李智慧

《李智慧 · 高并发架构实战课》 课程介绍 >

讲述：李智慧

时长 15:29 大小 14.18M

你好，我是李智慧。

网盘，又称云盘，是提供文件托管和文件上传、下载服务的网站（File hosting service）。人们通过网盘保管自己拍摄的照片、视频，通过网盘和他人共享文件，已经成为了一种习惯。我们准备开发一个自己的网盘应用系统，应用名称为“DBox”。

十几年前曾经有个段子，技术人员对老板说：您不能在公司电脑打开您家里电脑的文件，再贵的电脑也不能。事实上，随着网盘技术的成熟，段子中老板的需求已经成为现实：网盘可以自动将家里电脑的文件同步到公司电脑，老板可以在公司的电脑打开家里电脑的文件了。

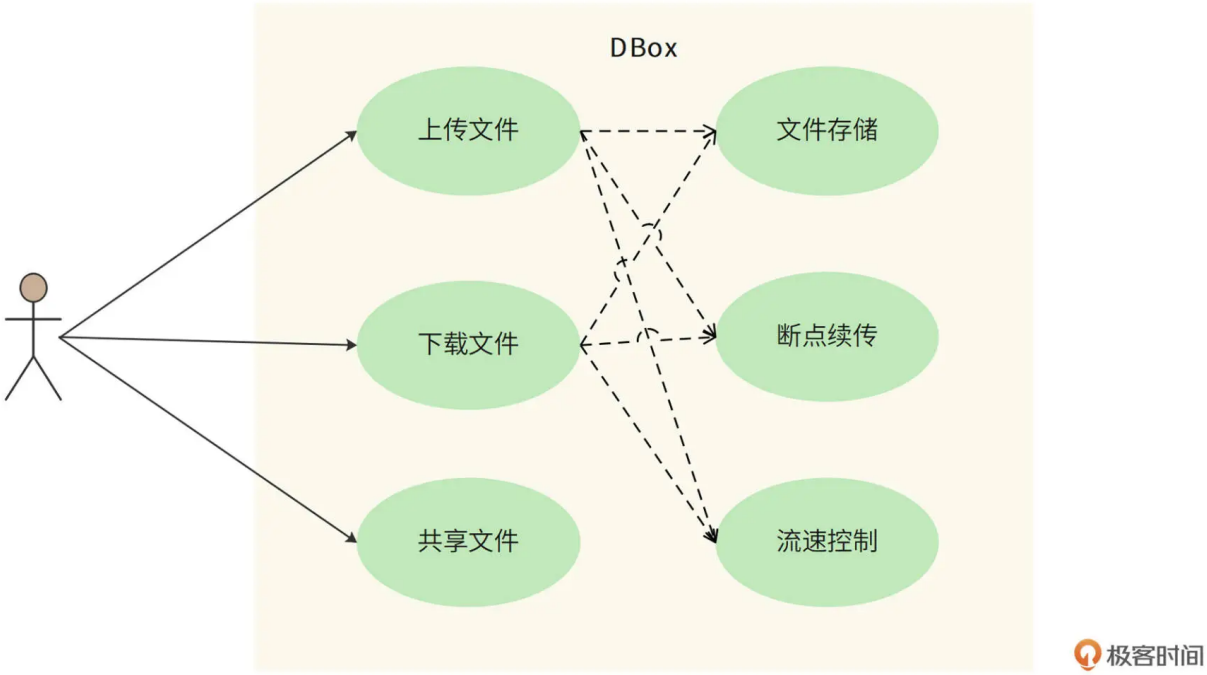
网盘的主要技术挑战是**海量数据的高并发读写访问。**用户上传的海量数据如何存储？如何避免部分用户频繁读写文件，消耗太多资源，而导致其他的用户体验不佳？我们看下 DBox 的技术架构以及如何解决这些问题。

需求分析

DBox 的核心功能是提供文件上传和下载服务。基于核心功能，DBox 需要在服务器端保存这些文件，并在下载和上传过程中实现断点续传。也就是说，如果上传或下载过程被中断了，恢复之后，还能从中断的地方重新上传或者下载，而不是从头再来。

DBox 还需要实现文件共享的需求。使用 DBox 的不同用户之间可以共享文件，一个用户上传的文件共享给其他用户后，其他用户也可以下载这个文件。

此外，网盘是一个存储和网络密集型的应用，用户文件占据大量**硬盘资源**，上传、下载需要占用大量网络带宽，并因此产生较高的运营成本。所以用户体验需要向付费用户倾斜，DBox 需要对上传和下载进行**流速控制**，保证付费用户得到更多的网络资源。DBox 用例图如下。



负载指标估算

DBox 的设计目标是支持 10 亿用户注册使用，免费用户最大可拥有 1TB 存储空间。预计日活用户占总用户的 20%，即 2 亿用户。每个活跃用户平均每天上传、下载 4 个文件。

DBox 的**存储量**、**吞吐量**、**带宽负载**估算如下。

总存储量

理论上，总存储空间估算为 10 亿 TB，即 1 万亿 GB。

$$10亿 \times 1TB = 10亿TB$$

但考虑到大多数用户并不会完全用掉这个空间，还有很多用户存储的文件其实是和别人重复的（电影、电子书、软件安装包等），真正需要的存储空间大约是这个估算值的 10%，即 1 亿 TB。

QPS

系统需要满足的平均 QPS 约为 10000。

$$2亿 \times 4 \div （24 \times 60 \times 60） \approx 1万$$

高峰期 QPS 约为平均 QPS 的两倍，即 2 万。

带宽负载

每次上传下载文件平均大小 1MB，所以需要网络带宽负载 10GB/s，即 80Gb/s。

$$1万 \times 1MB = 10GB/s = 80Gb/s$$

同样，高峰期带宽负载为 160Gb/s。

非功能需求

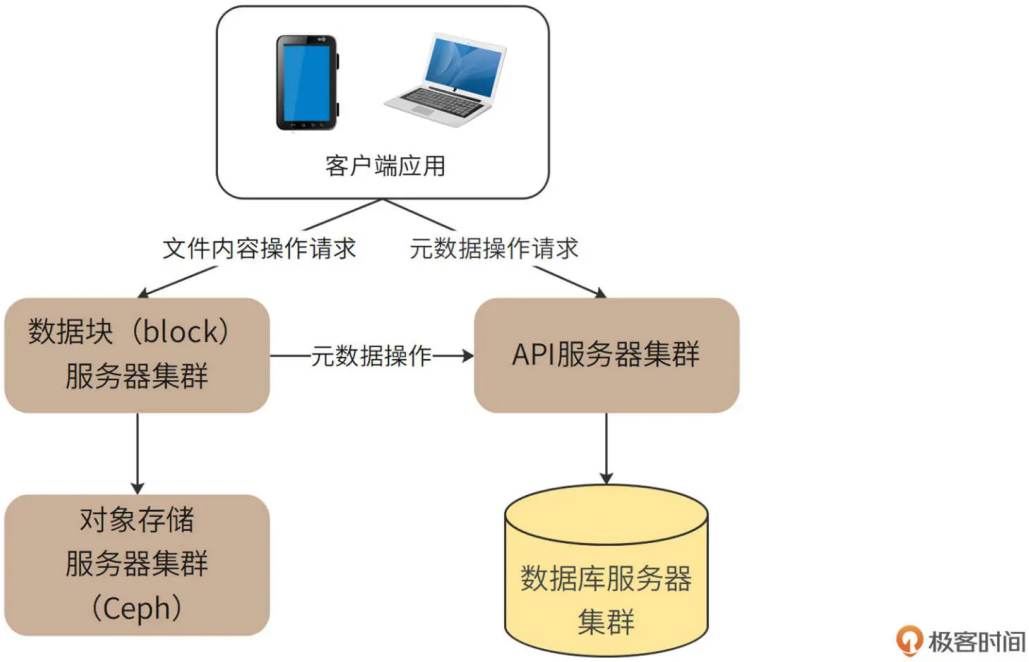
1. 大数据量存储：10 亿注册用户，1000 亿个文件，约 1 亿 TB 的存储空间。
2. 高并发访问：平均 1 万 QPS，高峰期 2 万 QPS。
3. 大流量负载：平均网络带宽负载 80Gb/S，高峰期带宽负载 160Gb/s。
4. 高可靠存储：文件不丢失，持久存储可靠性达到 99.9999% ，即 100 万个文件最多丢失（或损坏）1 个文件。
5. 高可用服务：用户正常上传、下载服务可用性在 99.99% 以上，即一年最多 53 分钟不可用。

6. 数据安全性：文件需要加密存储，用户本人及共享文件外，其他人不能查看文件内容。
7. 不重复上传：相同文件内容不重复上传，也就是说，如果用户上传的文件内容已经被其他用户上传过了，该用户不需要再上传一次文件内容，进而实现“秒传”功能。从用户视角来看，不到一秒就可以完成一个大文件的上传。

概要设计

网盘设计的关键是**元数据与文件内容的分离存储与管理**。所谓文件元数据就是文件所有者、文件属性、访问控制这些文件的基础信息，事实上，传统文件系统也是元数据与文件内容分离管理的，比如 Linux 的文件元数据记录在文件控制块 FCB 中，Windows 的文件元数据记录在文件分配表 FAB 中，Hadoop 分布式文件系统 HDFS 的元数据记录在 NameNode 中。

而 DBox 是将元信息存储在数据库中，文件内容则使用另外专门的存储体系。但是由于 DBox 是一个互联网应用，出于安全和访问管理的目的，并不适合由客户端直接访问存储元数据的数据库和存储文件内容的存储集群，而是通过 API 服务器集群和数据块服务器集群分别进行访问管理。整体架构如下图。

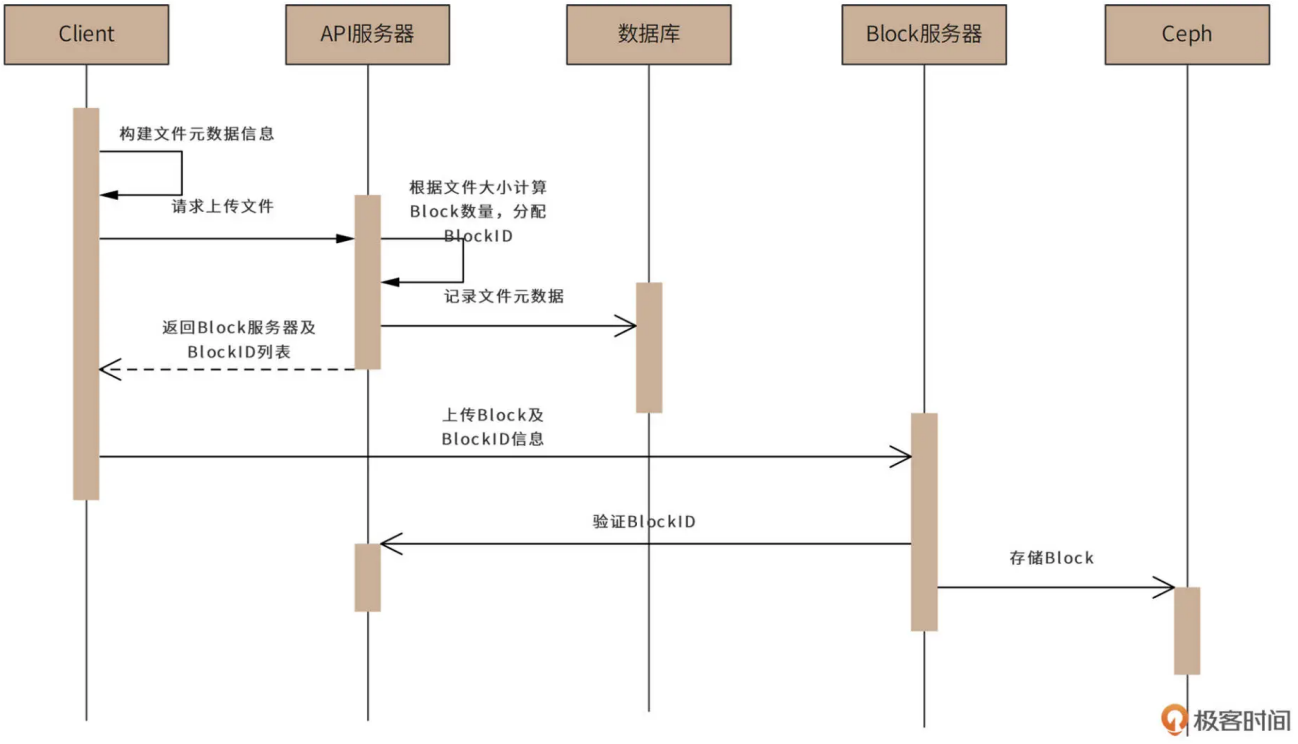


对于大文件，DBox 不会上传、存储一整个的文件，而是将这个文件进行切分，变成一个个单独的 Block，再将它们分别上传并存储起来。

这样做的核心原因是，DBox 采用对象存储作为最终的文件存储方案，而对象存储不适合存储大文件，需要进行切分。而大文件进行切分还带来其他的好处：可以**以 Block 为单位进行上传和下载，提高文件传输速度**；客户端或者网络故障导致文件传输失败，也只需要重新传输失败的 Block 就可以，进而实现**断点续传**功能。

Block 服务器就是负责 Block 上传和管理的。客户端应用程序根据 API 服务器的返回指令，将文件切分成一些 Block，然后将这些 Block 分别发送给 Block 服务器，Block 服务器再调用对象存储服务器集群，将 Block 存储在对象存储服务器中（DBox 选择 Ceph 作为对象存储）。

用户上传文件的时序图如下。

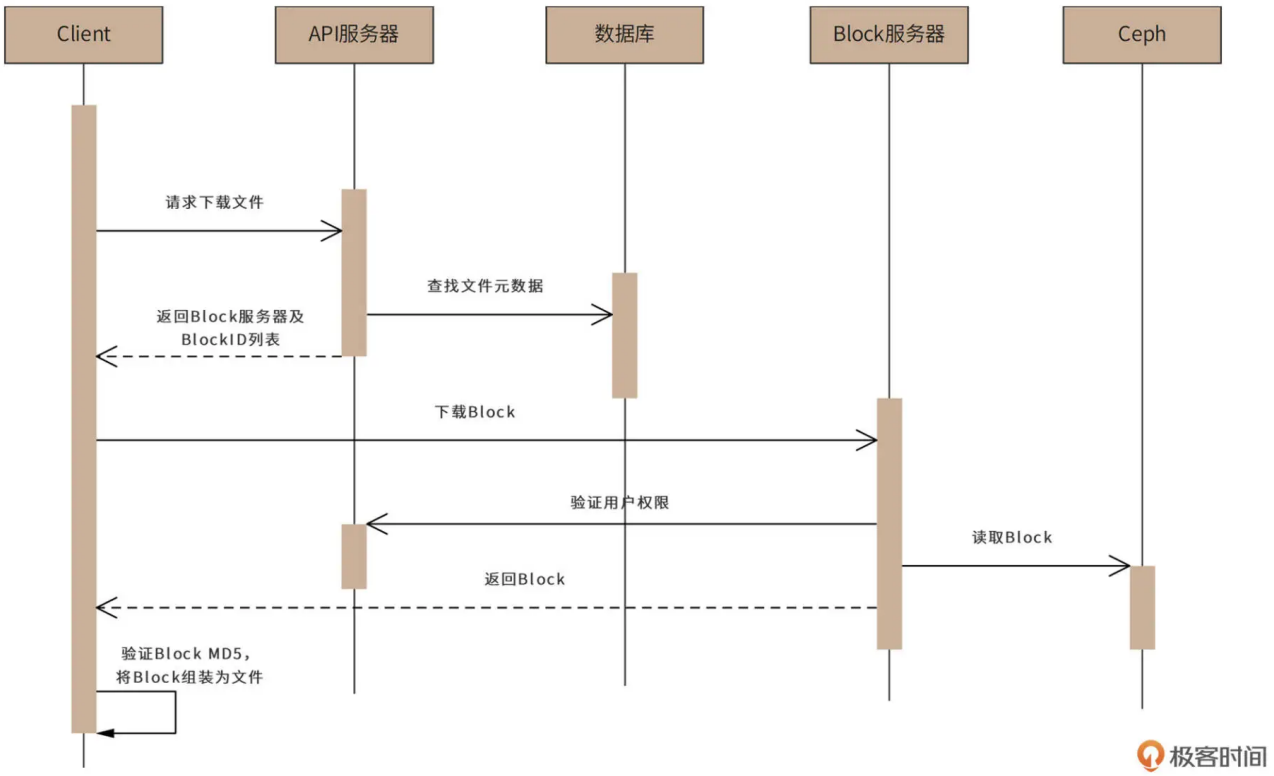


用户上传文件时，客户端应用程序收集文件元数据，包括文件名、文件内容 MD5、文件大小等等，并根据文件大小计算 Block 的数量（DBox 设定每个 block 大小 4MB），以及每个 Block 的 MD5 值。

然后客户端应用程序将全部元数据（包括所有 Block 的 MD5 值列表）发送给 API 服务器。API 服务器收到文件元数据后，为每个 Block 分配全局唯一的 BlockID（BlockID 为严格递增的 64 位正整数，总可记录数据大小 $2^{64} \times 4MB = 180亿PB$，足以满足 DBox 的应用场景）。

下一步，API 服务器将文件元数据与 BlockID 记录在数据库中，并将 BlockID 列表和应用程序可以连接的 Block 服务器列表返回客户端。客户端连接 Block 服务器请求上传 Block，Block 服务器连接 API 服务器进行权限和文件元数据验证。验证通过后，客户端上传 Block 数据，Block 服务器再次验证 Block 数据的 MD5 值，确认数据完整后，将 BlockID 和 Block 数据保存到对象存储集群 Ceph 中。

类似的，用户下载文件的时序图如下。



客户端程序访问 API 服务器，请求下载文件。然后 API 服务器会查找数据库，获得文件的元数据信息，再将元数据信息中的文件 BlockID 列表及可以访问的 Block 服务器列表返回给客户端。

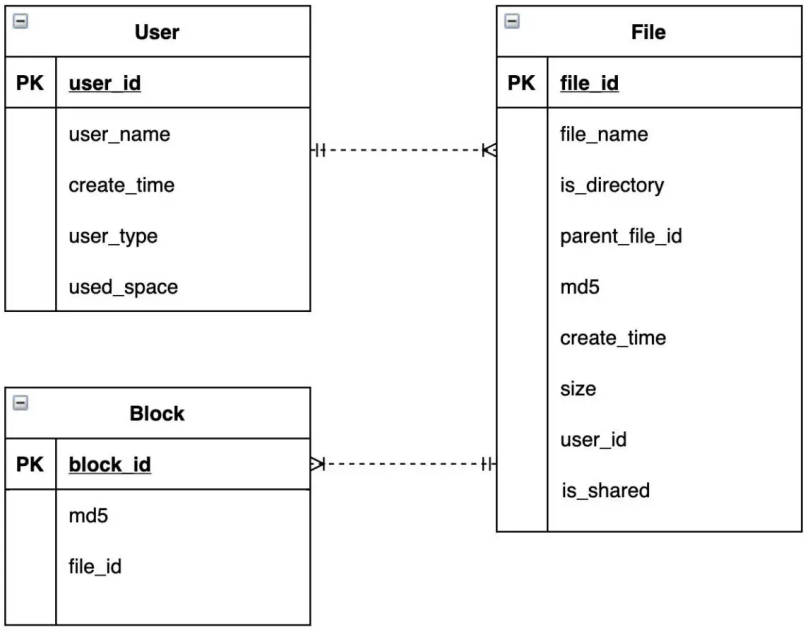
下一步，客户端访问 Block 服务器，请求下载 Block。Block 服务器验证用户权限后，从 Ceph 中读取 Block 数据，返回给客户端，客户端再将返回的 Block 组装为文件。

详细设计

为解决网盘的三个重要问题：元数据如何管理？网络资源如何向付费用户倾斜？如何做到不重复上传？DBox 详细设计将关注元数据库、上传下载限速、秒传的设计实现。

元数据库设计

元数据库表结构设计如下。



从图中可以看出，元数据库表结构中主要包括三个表，分别是 User 用户表、File 文件表和 Block 数据块表，表的用途和包含的主要字段如下：

1. User 用户表记录用户基本信息：用户名、创建时间、用户类型（免费、VIP）、用户已用空间、电话号码、头像等等。
2. File 文件表记录文件元信息：文件名、是否为文件夹、上级文件夹、文件 MD5、创建时间、文件大小、文件所属用户、是否为共享文件等。
3. Block 数据块表记录 Block 数据，包括 BlockID、Block MD5、对应文件等。

其中，User 表和 File 表为一对多的关系，File 表和 Block 表也是一对多的关系。

这 3 种表的记录数都是百亿级以上，所以元数据表采用**分片的关系数据库**存储。

因为查询的主要场景是根据用户 ID 查找用户信息和文件信息，以及根据文件 ID 查询 block 信息，所以 User 和 File 表都采用 user_id 作为分片键，Block 表采用 file_id 作为分片键。

限速

DBox 根据用户付费类型决定用户的上传、下载速度。而要控制上传、下载速度，可以通过限制并发 Block 服务器数目，以及限制 Block 服务器内的线程数来实现。

具体过程是，客户端程序访问 API 服务器，请求上传、下载文件的时候，API 服务器可以根据用户类型，决定分配的 Block 服务器数目和 Block 服务器内的服务线程数，以及每个线程的上传、下载速率。

Block 服务器会根据 API 服务器的返回值，来控制客户端能够同时上传、下载的 Block 数量以及传输速率，以此对不同用户进行限速。

秒传

秒传是用户快速上传文件的一种功能。

事实上，网盘保存的很多文件，内容其实是重复的，比如电影、电子书等等。一方面，重复上传这些文件会加大网盘的存储负载压力；另一方面，每次都要重新上传重复的内容，会导致用户网络带宽的浪费和用户等待时间过长的问题。

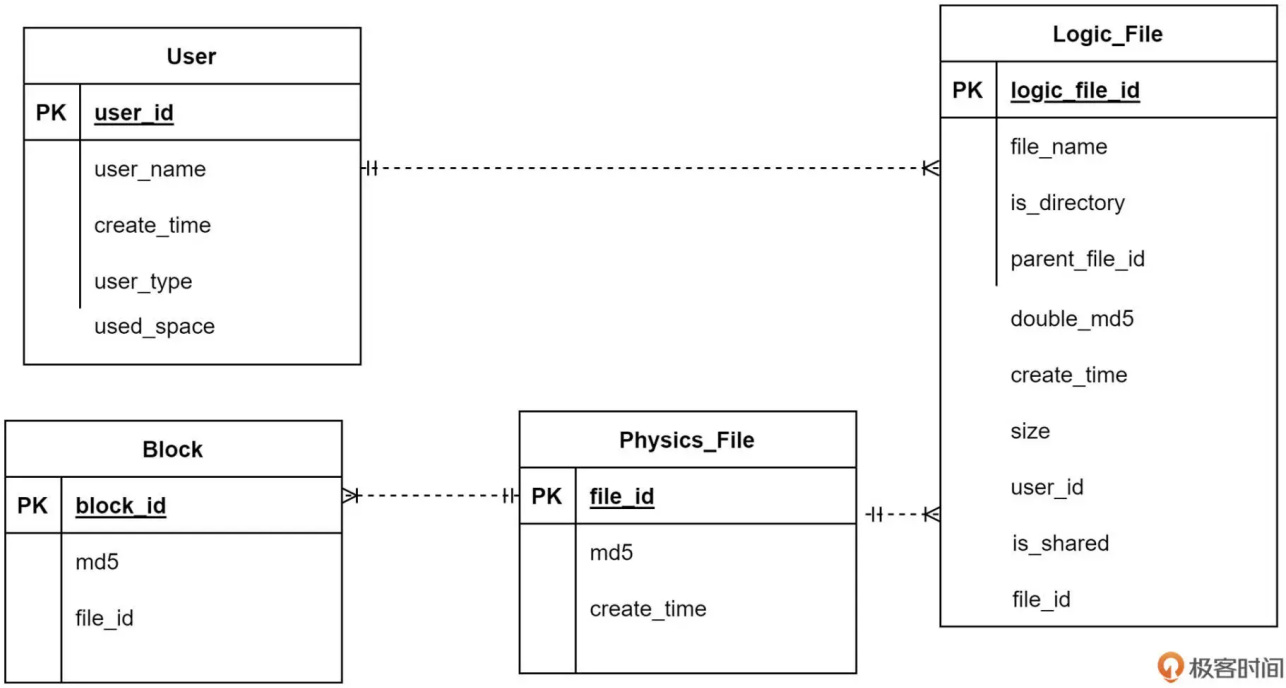
所以，在设计中，物理上相同的文件，DBox 只会保存一份。用户每次上传文件时，DBox 都会先在客户端计算文件的 MD5 值，再根据 MD5 值判断该文件是否已经存在。对于已经存在的文件，只需要建立用户文件和该物理文件的关联即可，并不需要用户真正上传该文件，这样就可以实现秒传的功能。

但是，计算 MD5 可能会发生 Hash 冲突，也就是不同文件算出来的 MD5 值是相同的，这样会导致 DBox 误判，将本不相同的文件关联到一个物理文件上。不但会使上传者丢失自己的文件，还会被黑客利用：上传一个和目标文件 MD5 相同的文件，然后就可以下载目标文件了。

所以，DBox 需要通过更多信息判断文件是否相同：**只有文件长度、文件开头 256KB 的 MD5 值、文件的 MD5 值，三个值都相同，才会认为文件相同**。当文件长度小于 256KB，则直接上传文件，不启用秒传功能。

为此，我们需要将上面的元数据库表结构进行一些改动，将原来的 File 表拆分成物理文件表 Physics_File 和逻辑文件表 Logic_File。其中，Logic_File 记录用户文件的元数据，并

和物理文件表 Physics_File 建立多对 1 关联关系，而 Block 表关联的则是 Physics_File 表，如下。



Logic_File 中字段 double_md5 记录了文件头 256KB 的 MD5、文件 MD5 两个数据拼接后的数据，而 size 记录了文件长度，只有这两个字段都相同才会启用秒传。

小结

我们在需求分析中讨论过，DBox 需要支持大数据量存储、高并发访问、高可用服务、高可靠存储等非功能需求。事实上，对于网盘应用而言，元数据 API 服务其实和一般的高并发互联网系统网关没有太大差别。真正有挑战的是海量文件的高可用存储，而这一挑战，在 DBox 中，被委托给了分布式对象存储 Ceph 来完成。而 Ceph 本身设计就是支持大数据量存储、高并发访问、高可用服务、高可靠存储的。

架构师按照职责，可以分成两种，一种是**应用系统架构师**，负责设计、开发类似网盘、爬虫这样的应用系统；另一种是**基础设施架构师**，负责设计、开发类似 Ceph、HDFS 这样的基础设施系统。

应用架构师需要掌握的技术栈更加**广泛**，要能够掌握各种基础设施技术的特性，并能根据业务特点选择最合适的方案；而基础设施架构师需要的技术栈更加**深入**，需要掌握计算机软硬件更深入的知识，才能开发出一个稳定的基础技术产品。

当然，最好的架构师应该是技术栈既广泛又深入，既能灵活应用各种基础设施来开发应用系统，也能在需要的时候自己动手开发新的基础设施系统。

我们专栏大部分案例都是关于应用的，但是也不乏关于编程框架、限流器、安全防火墙、区块链等基础设施的案例。你也可以在学习的过程中，感受下这两种系统的设计方案和技术关键点的不同。

思考题

网盘元数据存储采用分片的关系数据库方案，查询目录和文件都比较简单，但是性能也比较差。而且文件表按用户 ID 分片，如果某个用户创建大量文件，还会导致分片不均衡，你有什么优化的手段和方法吗？

欢迎在评论区分享你的思考，或者提出对这个设计文档的评审意见，我们共同进步。

分享给需要的人，Ta订阅超级会员，你将得 50 元

Ta单独购买本课程，你将得 20 元

生成海报并分享

赞 4 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 04 | 网页爬虫设计：如何下载千亿级网页？

下一篇 06 | 短视频系统设计：如何支持三千万用户同时在线看视频？

精选留言 (11)

写留言

peter

2022-02-25

请教老师几个问题：
Q1：用户数与并发数的估算。

一个500万注册用户的网站，并发数大约多少？如果业务是类似极客时间，带宽又需要多少？ 怎么估算并发数和带宽啊？
Q2：存储成本很高吗？...
展开

作者回复: 1 专栏2后半部分讨论过估算的思路，注册用户数->在线用户数->并发用户数，根据业务场景的不同，这个数据是逐级以几十倍到几百倍的比例递减，极客时间这个场景的话，500万注册用户，并发用户大概几千吧。

并发数 * 响应数据量 = 带宽

2 用户和存储是逐渐增加的，不是一致性投入1亿TB，稳定下来，每年实际投入百亿左右即可。这个成本不高的，百度网盘会员费每年300，我们十亿用户只要10%成为付费会员，营收300亿。

4 ceph是一个开源软件

2

3AM
2022-02-25

存储不用考虑多副本嘛？

作者回复: ceph已经实现多副本，所以网盘自己不需要考虑了

1

2022-03-02

课后题：可以尝试根据文件md5值进行分片。因为md5是无序的，所以应该可以使分片更加平均。

作者回复: 可是查找的时候，是按照用户和文件名查找的，用MD5分片如何完成查找呢？

Tim
2022-03-01

咨询下老师，用mysql存储这么大的数据是否合适？有没有其他的替代方案？

作者回复: 我也觉得不是很合适。

07提到了HBase替代方案。但是用HBase替代的话，存储结构如何设计呢？需要考虑下。

共 2 条评论 >

瑾年
2022-03-01

思考题：针对于数据倾斜的思考：
1、问题？数据倾斜导致资源不能充分利用；所以要围绕资源充分利用来处理
2、用户创建了大量文件，导致 File 和 Block 生成大量的数据，所以要实现File 和 Block 的均衡
...
展开

5G
2022-02-27

伏羲 Dbox 名字起的都好像华为云的内部工具

youngitachi
2022-02-26

思考题应该就是分库分表中可能遇到的数据倾斜的问题，这个时候可通过增加一个分片来解决，不过一般分库分表用的普通hash算法会涉及大量数据迁移，肯定是不行的，可以使用一致性hash算法来替代对分片key进行的普通的hash。
不过一般的一致性hash算法其实在新增实例的时候效果并不是特别好，因为它只会缓解该实例旁边的一个实例的存储压力，这个时候需要引入虚拟结点来处理。...
展开

作者回复: 嗯，一致性hash在查找文件或者block的时候如何查找呢？

youngitachi
2022-02-26

不知道是不是老师为了突出重点，有意在用例图里面没有提到用户查询功能，因此在后面计算QPS的时候也没有考虑，不过真正在设计这个系统的时候，应该还是需要考虑的吧？

作者回复: 篇幅有限，无法面面俱到

向东是大海
2022-02-26

系统设计很精妙，特别是秒传的设计！一个小建议，
MD5是已退役的哈希算法，采用SHA256等现行的哈希算法可减少哈希碰撞，更加安全。

作者回复: 谢谢建议

日月
2022-02-25

每个线程的上传、下载速率应该怎么控制呀

共 1 条评论 >

harryZ
2022-02-25

如果再多加一修建“净网功能”，找出盗版资源，大体是哪个实现思路？黑名单列表？还是？ 做成上传“和谐”的话可以在api中多加一个流程，那如果定期“和谐”，是否需要不断扫描文件元数据？及关了小黑屋后应怎办）

作者回复: 应该是多种手段一起使用：根据元数据识别违禁词、根据黑名单文件MD5进行匹配，对于文件内容通过机器学习识别。

元数据和MD5可以在API就处理，内容处理要后期扫描了。

共 2 条评论 >